SISG Module 3/SISMID Module 4

Introduction to R

Ken Rice
Tim Thornton

University of Washington

Seattle, July 2017

Introduction: Course Aims

This is a *first* course in R. We aim to cover;

- Reading in, summarizing & managing data
- Use of functions in R – doing jobs by programming, not by using drop-down menus (much)
- *Some* standard functions for statistical analysis – but minimal statistics in this module
- How to use other people's code, how to get help, what to learn next

We assume no previous use of R, also non-extensive programming skills in other languages. If this is *not* your level, please consider switching to a later module.

Introduction: Resources

Most importantly, the class site is

http://faculty.washington.edu/kenrice/rintro

Contains (or will contain);

- PDF copies of slides (in color, and contains a few hyperlinks)
- All datasets needed for exercises
- Exercises for you to try
- Our solutions to exercises (later!)
- Links to other software, other courses, book, and places to get R help
- Links to a few helpful websites/email list archives

Of course, search engines will find much more than this, and can be a useful start, when tackling analyses with R.

Introduction: About Tim

- Associate Prof, UW Biostat
- A useR and an instructoR
- Research in Genetic Epidemiology for Complex Human Traits

Introduction: About Ken

- Associate Prof, UW Biostat
- AuthoR of a few R packages, useR, teacheR
- Genetic/Genomic research in Cardiovascular Epidemiology

... and you?

(Briefly, who are you, what's your genetics/infectious disease?)

Introduction: Course structure

10 sessions over 2.5 days

- Day 1; (Mostly RStudio) Data management, using functions
- Day 2; (Standard R) More about programming
- Day 2.5; More advanced ideas

Web page: `http://faculty.washington.edu/kenrice/rintro/`

Introduction: Session structure

What to expect in a typical session;

- 45 mins teaching (please interrupt!)
- 30 mins hands-on; please work in pairs
- 15 mins summary, discussion/extensions (interrupt again!)

There will also be one ‘take-home’ exercise, on Day 2; the final session will include in-depth discussion/evaluation.

Please note: the 2.5 day course moves quickly, and later material builds on earlier material. So, **please interrupt!**

1. Reading in data

Ken Rice
Tim Thornton

University of Washington

Seattle, July 2017

What is R?

R is a ‘programming environment for statistics and graphics’

- Does *basically* everything, can also be extended
- It’s the default when statisticians implement new methods
- Free, open-source

But;

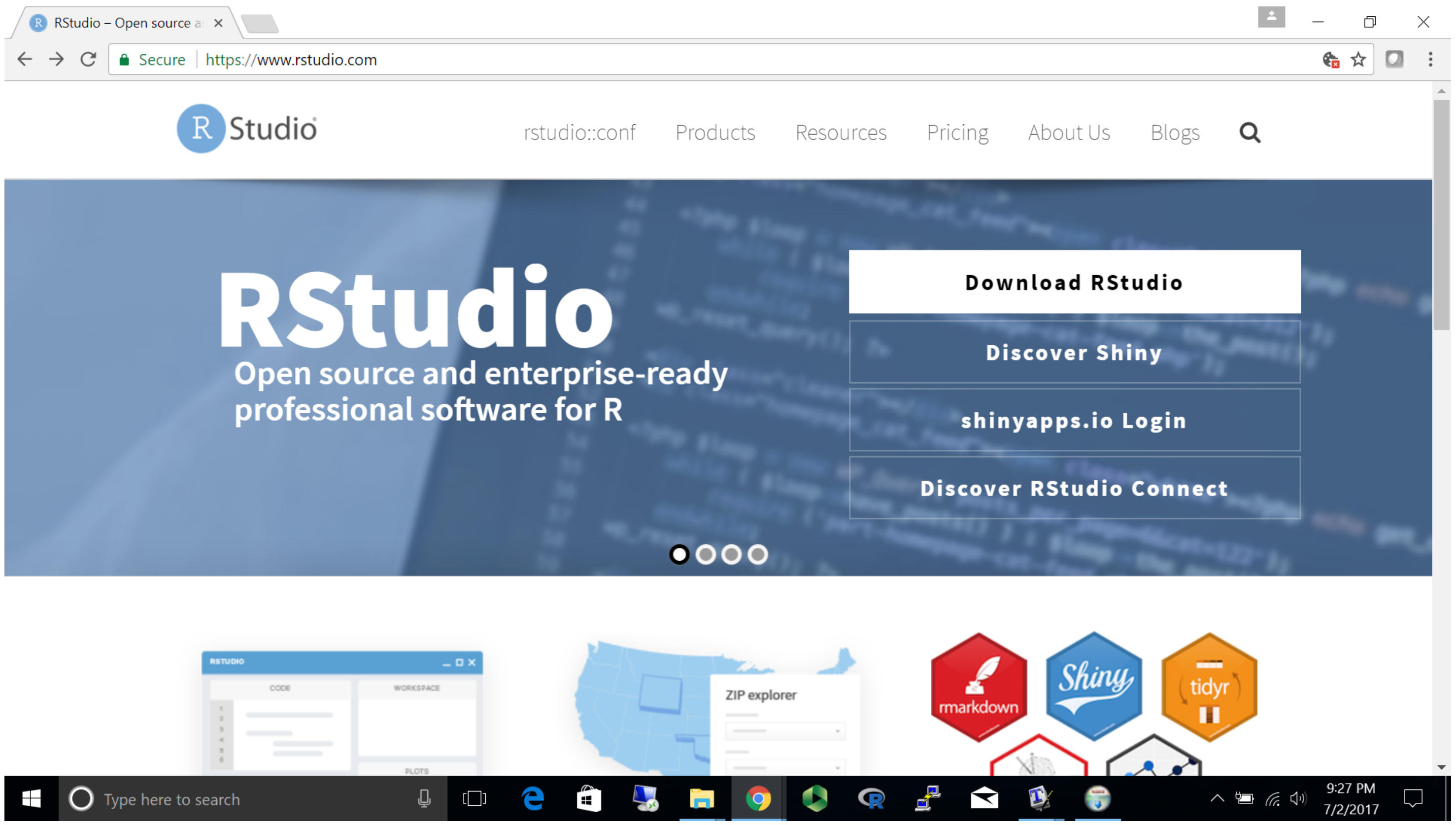
- Steeper learning curve than e.g. Excel, Stata
- Command-line driven (programming, not drop-down menus)
- Gives only what you ask for!

To help with these difficulties, we will begin with RStudio, a graphical user interface (front-end) for R that is slightly more user-friendly than ‘Classic’ R’s GUI.

So *after* installing the latest version of R...

RStudio

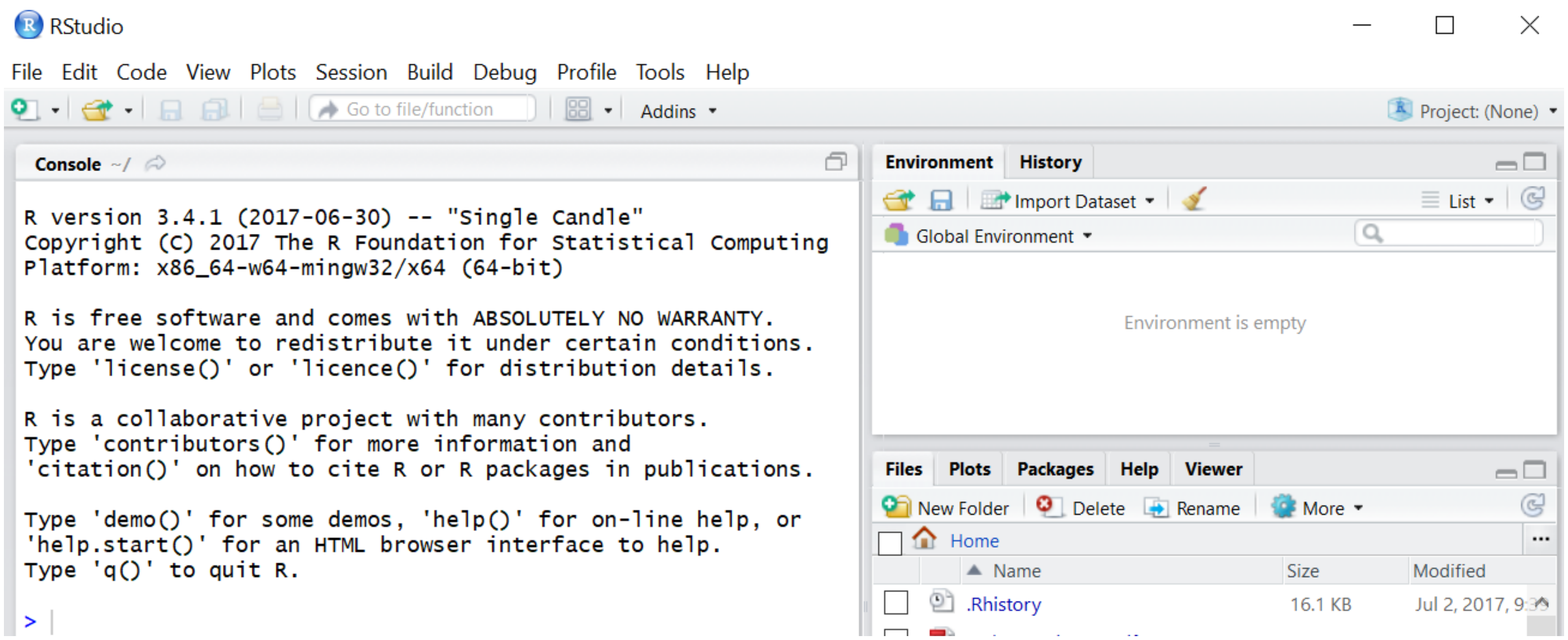
In your favorite web browser, download from rstudio.com;



- Select& download the FREE installer for *your* system
- Default installation is fine
- Working in pairs *highly* recommended

RStudio

You should also install R - RStudio is a ‘front end’ to R itself. On first startup, RStudio should look like this; (up to version and Mac/PC differences)


RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
Console ~/
Environment History
Import Dataset
List
Global Environment
Environment is empty
Files Plots Packages Help Viewer
New Folder Delete Rename More
Home
Name Size Modified
.Rhistory 16.1 KB Jul 2, 2017, 9:3

```
R version 3.4.1 (2017-06-30) -- "Single Candle"
Copyright (C) 2017 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

If you’ve used it before, RStudio defaults to remembering what you were doing.

RStudio

We'll use the 'Console' window first – as a (fancy!) calculator

```
> 2+2
[1] 4
> 2^5+7
[1] 39
> 2^(5+7)
[1] 4096
> exp(pi)-pi
[1] 19.9991
> log(20+pi)
[1] 3.141632
> 0.05/1E6   # a comment; note 1E6 = 1,000,000
[1] 5e-08
```

- All common math functions are available; parentheses (round brackets) work as per high school math
- Try to get used to bracket matching. A '+' prompt means the line isn't finished – hit Escape to get out, then try again.

RStudio

R stores data (and everything else) as *objects*. New objects are created when we *assign* them values;

```
> x <- 3
> y <- 2 # now check the Environment window
> x+y
[1] 5
```

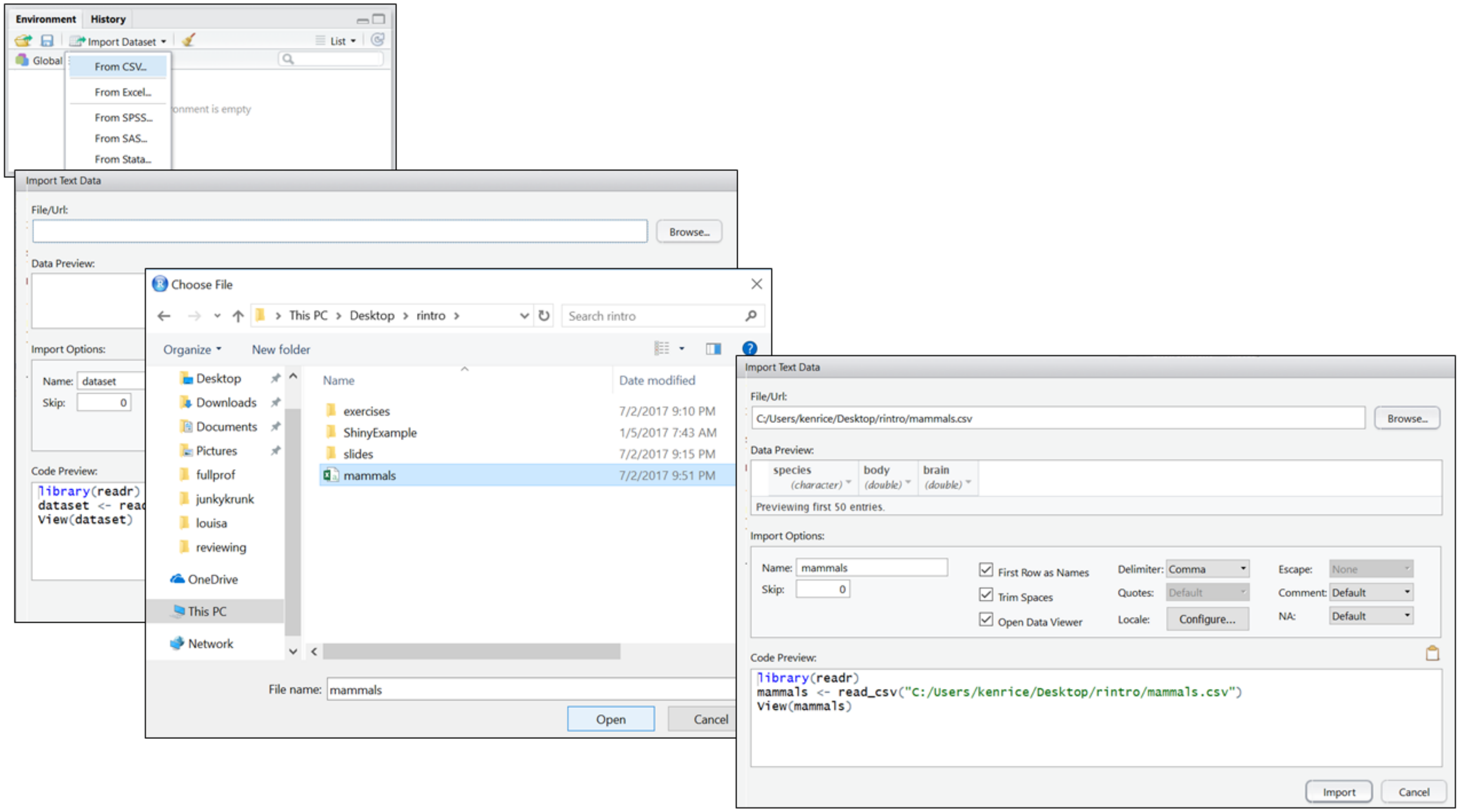
Assigning new values to existing objects over-writes the old version – and be aware there is no Ctrl-Z ‘undo’;

```
> y <- 17.4 # check the Environment window again
> x+y
[1] 20.4
```

- Anything after a hash (#) is ignored – e.g. comments
- Spaces don’t matter
- Capital letters *do* matter

RStudio: Reading in data

To import a dataset, follow pop-ups from the Environment tab;



Import Text Data

File/Url:

Data Preview:

Import Options:

Name: dataset
Skip: 0

Code Preview:

```
library(readr)
dataset <- read
View(dataset)
```

Name	Date modified
exercises	7/2/2017 9:10 PM
ShinyExample	1/5/2017 7:43 AM
slides	7/2/2017 9:15 PM
mammals	7/2/2017 9:51 PM

File name: mammals

Import Text Data

File/Url: C:/Users/kenrice/Desktop/rintro/mammals.csv

Data Preview:

species (character)	body (double)	brain (double)

Previewing first 50 entries.

Import Options:

Name: mammals
Skip: 0
☑ First Row as Names
☑ Trim Spaces
☑ Open Data Viewer
Delimiter: Comma
Quotes: Default
Locale: Configure...
Escape: None
Comment: Default
NA: Default

Code Preview:

```
library(readr)
mammals <- read_csv("C:/Users/kenrice/Desktop/rintro/mammals.csv")
View(mammals)
```

RStudio: Reading in data

More on those options;

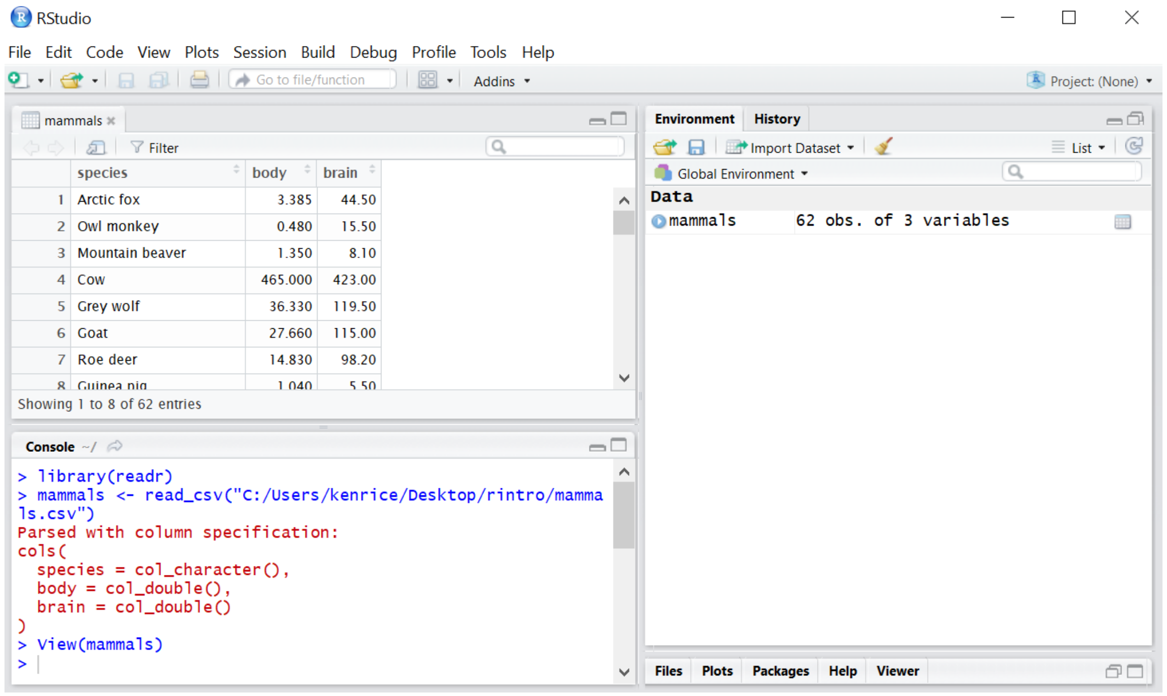
- **Name**: Name of the object that will store the whole dataset, when it's read in
- **First row as names**: names make variables easier to understand
- **Delimiter**: what's between items on a single line?
- **NA**: How are missing values denoted?

The defaults are sensible, but R assumes you *know* what your data *should* look like – and whether it has named columns, numeric/character data, etc. *No* software is smart enough to cope with every format that might be used by you/your colleagues to store data.

RStudio: Reading in data

After successfully reading in the data;

- The environment now includes a `mammals` object – or whatever you called the data read from file
- A copy of the data can be examined in the Excel-like data viewer (below) – if it looks weird, find out why & fix it!



	species	body	brain
1	Arctic fox	3.385	44.50
2	Owl monkey	0.480	15.50
3	Mountain beaver	1.350	8.10
4	Cow	465.000	423.00
5	Grey wolf	36.330	119.50
6	Goat	27.660	115.00
7	Roe deer	14.830	98.20
8	Guinea pig	1.040	5.50

Showing 1 to 8 of 62 entries

```
Data
mammals       62 obs. of 3 variables
```

```
> library(readr)
> mammals <- read_csv("C:/Users/kenrice/Desktop/rintro/mamma
ls.csv")
Parsed with column specification:
cols(
  species = col_character(),
  body = col_double(),
  brain = col_double()
)
> View(mammals)
> 
```

... we'll return later, to `read_csv()` in the Console window

RStudio: Reading in data

What's a good name for my new object?

- Something memorable (!) and not easily-confused with other objects, e.g. `X` isn't a good choice if you already have `x`
- Names must start with a letter or period ("."), after that any letter, number or period is okay
- Avoid other characters; they get interpreted as math ("-","*") or are hard to read ("_") so should not be used in names
- Avoid names of existing functions – e.g. `summary`. Some one-letter choices (`c`, `C`, `F`, `t`, `T` and `S`) are already used by R as names of functions, it's best to avoid these too

Operating on data

To operate on data, type commands in the Console window, just like our earlier calculator-style approach;

```
> str(mammals) # edited output
Classes tbl_df, tbl and 'data.frame': 62 obs. of  3 variables:
 $ species: chr  "Arctic fox" "Owl monkey" "Mountain beaver" "Cow" ...
 $ body   : num  3.38 0.48 1.35 465 36.33 ...
 $ brain  : num  44.5 15.5 8.1 423 119.5 ...
> summary(mammals)
   species                body                brain
 Length:62          Min.    :   0.005   Min.    :   0.14
 Class :character   1st Qu.:   0.600   1st Qu.:   4.25
 Mode  :character   Median :   3.342   Median :  17.25
                    Mean   : 198.790   Mean   : 283.13
                    3rd Qu.:  48.203   3rd Qu.: 166.00
                    Max.   :6654.000   Max.   :5712.00
```

- `str()` tells us the `str`ucture of an object
- `summary()` summarizes the object

Can also use these commands on any object – e.g. the single numbers we created earlier (try it!)

Operating on data: columns

Individual columns in data frames are identified using the $ symbol – just seen in the `str()` output.

```
> mammals$brain
 [1]   44.50   15.50    8.10  423.00  119.50  115.00   98.20    5.50   58.00
[10]    6.40    4.00    5.70    6.60    0.14    1.00   10.80   12.30    6.30
[19] 4603.00    0.30  419.00  655.00    3.50  115.00   25.60    5.00   17.50
[28]  680.00  406.00  325.00   12.30 1320.00 5712.00    3.90  179.00   56.00
[37]   17.00    1.00    0.40    0.25   12.50  490.00   12.10  175.00  157.00
[46]  440.00  179.50    2.40   81.00   21.00   39.20    1.90    1.20    3.00
[55]    0.33  180.00   25.00  169.00    2.60   11.40    2.50   50.40
> summary(mammals$brain)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.14    4.25   17.25  283.10  166.00 5712.00
```

Think of $ as 'apostrophe-S', i.e. `mammals`'S `brain`.

Unlike many other statistical packages, R can handle *multiple* datasets at the same time – helpful if your data are e.g. phenotypes & genotypes, or county & disease outbreak data. This isn't possible without $, or *some* equivalent syntax.

Operating on data: columns

New columns are created when you assign their values – here containing the brain weights in kilograms;

```
> mammals$brainkg <- mammals$brain/1000
> str(mammals)
Classes tbl_df, tbl and 'data.frame': 62 obs. of  4 variables:
 $ species: chr  "Arctic fox" "Owl monkey" "Mountain beaver" "Cow" ...
 $ body   : num  3.38 0.48 1.35 465 36.33 ...
 $ brain  : num  44.5 15.5 8.1 423 119.5 ...
 $ brainkg: num  0.0445 0.0155 0.0081 0.423 0.1195 ...
> summary(mammals$brainkg)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.00014 0.00425 0.01725 0.28313 0.16600 5.71200
```

- Assigning values to existing columns over-writes existing values – again, with no warning
- With e.g. `mammals$newcolumn <- 0`, the new column has every entry zero; R *recycles* this single value, for every entry
- It's unusual to delete columns... but if you *must*;
  `mammals$brainkg <- NULL`

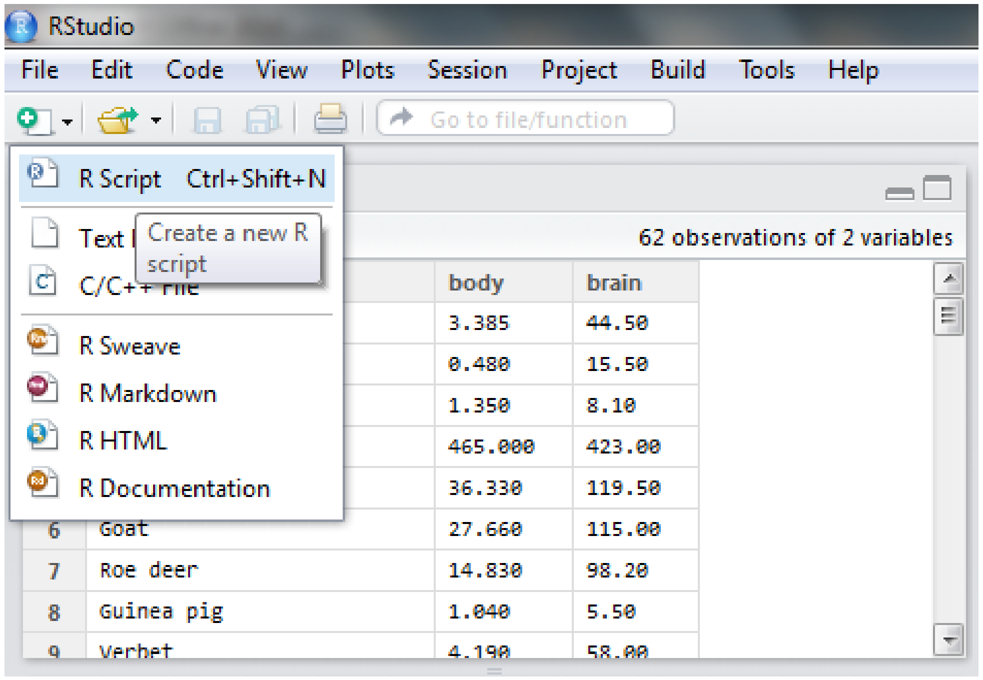
Operating on data: columns

Some other functions useful for summarizing data frames, and their columns;

```
> names(mammals)
[1] "species" "body"    "brain"
> dim(mammals)      # dim is short for dimension
[1] 62  3
> length(mammals$body) # how many rows in our dataset?
[1] 62
> min(mammals$body)
[1] 0.005
> max(mammals$body)
[1] 6654
> range(mammals$body)
[1]    0.005 6654.000
> mean(mammals$brain)
[1] 283.1342
> sd(mammals$brain) # sd is short for standard deviation
[1] 930.2789
> median(mammals$brain)
[1] 17.25
```

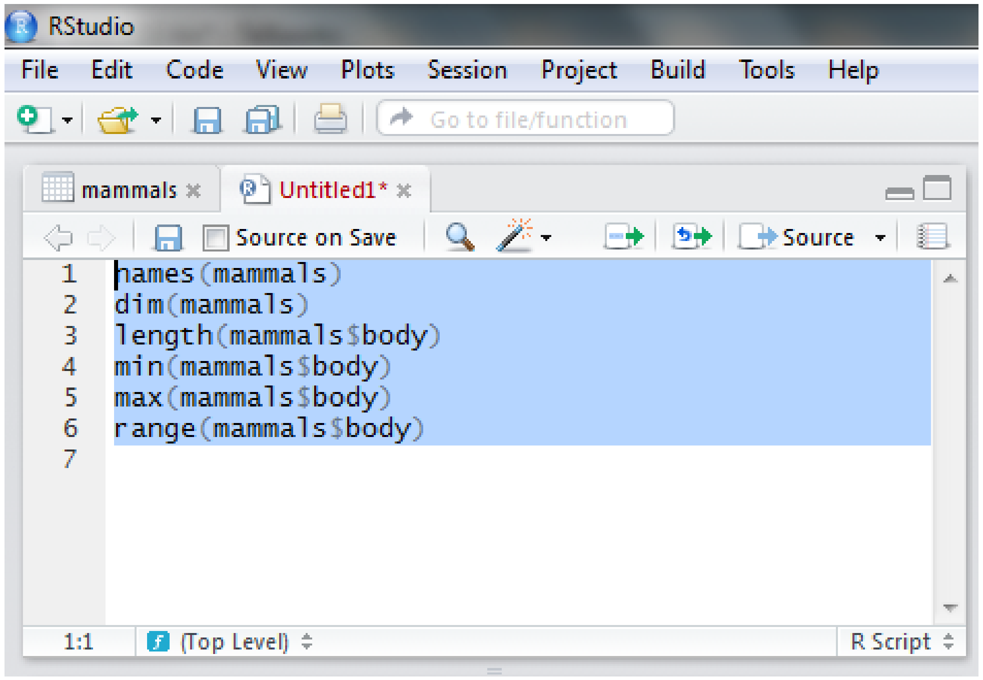
RStudio: the Script window

While fine for occasional use, entering *every* command 'by hand' is error-prone, and quickly gets tedious. A *much* better approach is to use a Script window – open one with Ctrl-Shift-N, or the drop-down menus;



62 observations of 2 variables

		body	brain
		3.385	44.50
		0.480	15.50
		1.350	8.10
		465.000	423.00
		36.330	119.50
6	Goat	27.660	115.00
7	Roe deer	14.830	98.20
8	Guinea pig	1.040	5.50
9	Verbet	4.190	58.00



```
names(mammals)
dim(mammals)
length(mammals$body)
min(mammals$body)
max(mammals$body)
range(mammals$body)
```

- Opens a nice editor, enables saving code (.R extension)
- Run current line (or selected lines) with Ctrl-Enter, or Ctrl-R

RStudio: the Script window

An important notice: from now on, we assume you are using a script editor.

- First-time users tend to be reluctant to switch! – but it's worth it, ask any experienced user
- Some code in slides may be formatted for cut-and-paste into scripts – it may not look exactly like what appears in the Console window
- Exercise 'solutions' given as .R files
- Scripts make it easy to run slightly modified code, without re-typing everything – remember to save them as you work
- Also remember the Escape key, if e.g. your bracket-matching goes wrong

For a very few jobs, e.g. changing directories, we'll still use drop-down menus. But commands *are* available, for all tasks.

Operating on data: subsets

To identify general subsets – not just the columns selected by $ – R uses square brackets. Selecting individuals elements;

```
> mammals$brain[32] # 32nd element of mammals$brain
[1] 1320
> mammals$brain[32] # 32nd element of mammals$brain
[1] 1320
> mammals$species[32]
[1] "Human"
> mammals$body[32]
[1] 62
```

Can also select entire columns or entire rows this way – and ‘blank’ entries indicate you want everything.

```
> mammals[32,2] # subtable with just 32nd row, 2nd column
# A tibble: 1 x 1
   body
1    62
> mammals[32,]      # everything in the 32nd row
# A tibble: 1 x 3
  species  body brain
1   Human    62  1320
> sum(mammals[,3]) # sum of all the brains masses...
[1] 17554.32
```

Operating on data: subsets

Suppose we were interested in the brain weight (i.e. 2nd column) for mammals (i.e. rows) 14, 55, & 61. How to select these multiple elements?

```
> mammals[c(14,55,61),2]
   body
1 0.005 # check these against data view
2 0.048
3 0.104
```

But what is `c(14,55,61)`? It's a *vector* of numbers – `c()` is for *combine*;

```
> length(c(14,55,61))
[1] 3
> str(c(14,55,61))
 num [1:3] 14 55 61
```

We can select these rows and all the columns;

```
> mammals[c(14,55,61),]
                    species  body brain
1 Lesser short-tailed shrew 0.005  0.14
2                Musk shrew 0.048  0.33
3                Tree shrew 0.104  2.50
```

Operating on data: subsets

A very useful special form of vector;

```
> 1:10
 [1]  1  2  3  4  5  6  7  8  9 10
> 6:2
[1] 6 5 4 3 2
> -1:-3
[1] -1 -2 -3
```

R expects you to know this shorthand – see e.g. its use of `1:3` in the output from `str()`, on the previous slide. For a ‘rectangular’ selection of rows and columns;

```
> mammals[20:22, 1:3]
        species    body brain
1 Big brown bat   0.023   0.3
2        Donkey 187.100 419.0
3         Horse 521.000 655.0
```

Negative values correspond to *dropping* those rows/columns;

```
> mammals[-3:-62, c(1,3)] # everything but the first two rows, & columns c(1,3)
     species brain
1 Arctic fox  44.5
2 Owl monkey  15.5
```

Operating on data: subsets

As well as storing numbers and character strings (like `"Donkey"`, `"Big brown bat"`) R can also store *logicals* – TRUE and FALSE.

To make a new vector, with elements that are TRUE if body mass is above 500kg and FALSE otherwise;

```
> is.heavy <- mammals$body > 500
> table(is.heavy)  # another useful data summary command
is.heavy
FALSE  TRUE
   58     4
```

Which mammals were these? (And what were their masses?)

```
> mammals[is.heavy,]          # just the rows for which is.heavy is TRUE
           species  body brain
1   Asian elephant  2547  4603
2            Horse   521   655
3          Giraffe   529   680
4 African elephant  6654  5712
```

Use e.g. `mammals[is.heavy,3]` to combine TRUE/FALSE (rows) and numbers (columns)

Operating on data: subsets

One final method... for now! Instead of specifying columns of interest by number, or through vectors of TRUEs/FALSEs, we can also just give the names – as *character strings*, or vectors of character strings.

```
> mammals[1:3,"body"]
   body
1 3.385
2 0.480
3 1.350
> mammals[is.heavy,c("species","body")]
           species  body
1   Asian elephant  2547
2            Horse   521
3          Giraffe   529
4 African elephant  6654
```

– this is more typing than the other methods, but is *much* easier to debug/reuse. Neither is 'right' or 'wrong' – R is just flexible.

Quitting time (almost)

When you're finished with RStudio;

- Ctrl-Q, or the drop-down menus, or entering `q()` at the command line all start the exit process
- You will be asked "Save workspace image to ~/.RData?"
  - No/Don't Save: nothing is saved, and is not available when you re-start. *This is recommended*, because you will do different things in each session
  - Yes: Everything in memory is stored in R's internal format (.Rdata) and will be available when you re-start RStudio
  - Cancel: don't quit, go back
- Writing about what you did (output from a script) often takes much longer than re-running that script's analyses – so often, a 'commented' script is all the R you need to store

To get rid of *objects* in your current session, use `rm()`, e.g. `rm(is.heavy, mammals, x, y)` ... or RStudio's 'broom' button.

Summary

- In RStudio, read in data from the pop-up menu in the Environment window (or Tools menu)
- Data frames store data; can have many of these objects – and multiple other objects, too*
- Identify vectors with $, subsets with square brackets
- Many useful summary functions are available, with sensible names
- Scripts are an important drudgery-avoidance tool!

* ... RStudio's new-ish 'tibble' format is actually a very close relative of a standard R data frame. Not all sources take this into account.